

This Page Is Inserted by IFW Operations
and is not a part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

**As rescanning documents *will not* correct images,
please do not report the images to the
Image Problem Mailbox.**



PATENT ABSTRACTS OF JAPAN

(11) Publication number: 06012467 A

(43) Date of publication of application: 21 . 01 . 94

(51) Int. Cl

G06F 15/60

(21) Application number: 04167727

(71) Applicant: SONY CORP

(22) Date of filing: 25 . 06 . 92

(72) Inventor: OKI MITSUHARU

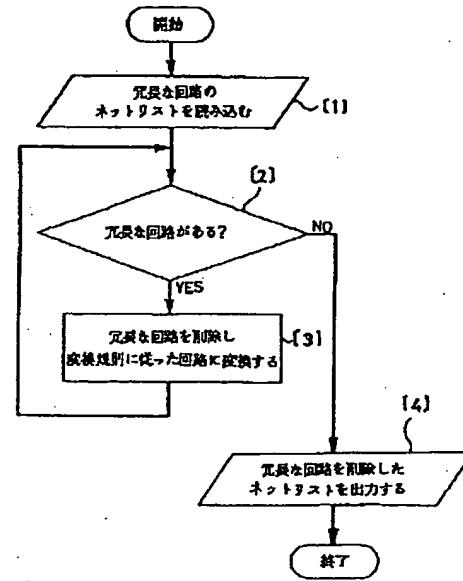
(54) LOGIC CIRCUIT COMPRESSING METHOD

(57) Abstract:

PURPOSE: To realize an LSI having a small circuit scale by deleting redundant circuits from a network list of redundant circuit constitutions to generate a new network list and designing layout from this new network list.

CONSTITUTION: In a step 1, the network list of redundant circuits is read in. In a step 2, it is discriminated whether a specific redundant circuit exists in the network list or not. If it exists there, the redundant circuit is deleted in accordance with a conversion rule in a step 3. Steps 2 and 3 are repeated until redundant circuits are completely deleted from the network list. When redundant circuits are completely deleted from the network list in the step 2, the new list of circuits other than redundant circuits is outputted in a step 4.

COPYRIGHT: (C)1994,JPO&Japio



(51)Int.Cl.⁵
G 0 6 F 15/60識別記号 庁内整理番号
3 6 0 K 7922-5L

F I

技術表示箇所

審査請求 未請求 請求項の数7(全14頁)

(21)出願番号 特願平4-167727
 (22)出願日 平成4年(1992)6月25日

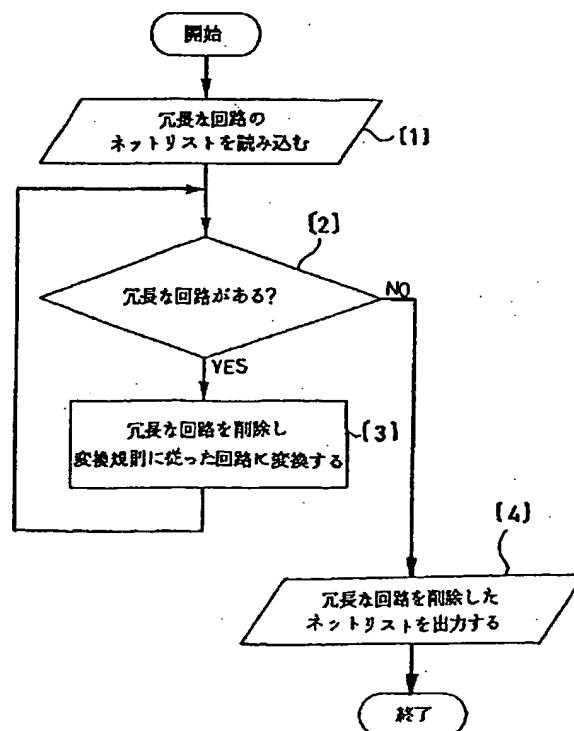
(71)出願人 000002185
 ソニー株式会社
 東京都品川区北品川6丁目7番35号
 (72)発明者 大木 光晴
 東京都品川区北品川6丁目7番35号 ソニー株式会社内
 (74)代理人 弁理士 松隈 秀盛

(54)【発明の名称】論理回路圧縮方法

(57)【要約】

【目的】冗長な回路構成のネットリストから冗長な回路を削除したネットリストを作り、このような新たなネットリストからレイアウト設計することにより回路規模の小さいLSIを実現する。

【構成】ステップ〔1〕で冗長な回路のネットリストが読み込まれる。次にステップ〔2〕でそのネットリスト内に特定の冗長な回路があるかないかが判断される。ここでもし冗長な回路があればステップ〔3〕で変換規則に従って冗長な回路が削除される。さらにこのステップ〔2〕及び〔3〕が冗長な回路がなくなるまで繰り返し実行される。そしてステップ〔2〕でそのネットリスト内にある冗長な回路がなくなると、ステップ〔4〕で冗長でない回路のネットリストが出力される。



【特許請求の範囲】

【請求項1】少なくとも1つの入力信号は定数である加算器若しくは単位遅延素子を含む冗長な回路構成のネットリストを入力とし、

特定の変換規則によって冗長でない回路構成のネットリストを出力するようにしたことを特徴とする論理回路圧縮方法。

【請求項2】上記特定の変換規則は、

3つの入力とも常に0である全加算器は削除し、全加算器の出力であるキャリアアウト端子の配線に0を供給させ、全加算器の出力であるサム端子の配線に0を供給させる、

2つの入力が常に0であり、残る1つの入力が常に1である全加算器は削除し、全加算器の出力であるキャリアアウト端子の配線に0を供給させ、全加算器の出力であるサム端子の配線に1を供給させる、

1つの入力が常に0であり、残る2つの入力が常に1である全加算器は削除し、全加算器の出力であるキャリアアウト端子の配線に1を供給させ、全加算器の出力であるサム端子の配線に0を供給させる、

3つの入力とも常に1である全加算器は削除し、全加算器の出力であるキャリアアウト端子の配線に1を供給させ、全加算器の出力であるサム端子の配線に1を供給させる、という変換の少なくとも一つを含んでいることを特徴とする請求項1に記載の論理回路圧縮方法。

【請求項3】上記特定の変換規則は、

2つの入力が常に0であり、残る1つの入力が乗数、被乗数の値によって変化する全加算器は削除し、全加算器の出力であるキャリアアウト端子の配線に0を供給させ、全加算器の出力であるサム端子の配線に上記乗数、被乗数の値によって変化する値を供給させる、

1つの入力が常に0であり、もう1つの入力が常に1であり、残る1つの入力が乗数、被乗数の値によって変化する全加算器は削除し、全加算器の出力であるキャリアアウト端子の配線に上記乗数、被乗数の値によって変化する値を供給させ、全加算器の出力であるサム端子の配線に上記乗数、被乗数の値によって変化する値の反転値を供給させる、

2つの入力が常に1であり、残る1つの入力が乗数、被乗数の値によって変化する全加算器は削除し、全加算器の出力であるキャリアアウト端子の配線に1を供給させ、全加算器の出力であるサム端子の配線に上記乗数、被乗数の値によって変化する値を供給せる、という変換の少なくとも一つを含んでいることを特徴とする請求項1に記載の論理回路圧縮方法。

【請求項4】上記特定の変換規則は、

1つの入力が常に0であり、残る2つの入力が乗数、被乗数の値によって変化する全加算器は削除し、半加算器の2つの入力端子に上記乗数、被乗数の値によって変化する2つの値を供給させ、全加算器の出力であるキャリ

アウト端子の配線に上記半加算器の出力であるキャリアアウトを供給させ、全加算器の出力であるサム端子の配線に上記半加算器の出力であるサムを供給させる、

1つの入力が常に1であり、残る2つの入力が乗数、被乗数の値によって変化する全加算器は削除し、全加算器の出力であるキャリアアウト端子の配線に上記乗数、被乗数の値によって変化する2つの値の論理和を供給させ、全加算器の出力であるサム端子の配線に上記乗数、被乗数の値によって変化する2つの値の排他的否定論理和を供給させる、という変換の少なくとも一つを含んでいることを特徴とする請求項1に記載の論理回路圧縮方法。

【請求項5】上記特定の変換規則は、

2つの入力とも常に0である半加算器は削除し、半加算器の出力であるキャリアアウト端子の配線に0を供給させ、半加算器の出力であるサム端子の配線に0を供給させる、

1つの入力が常に0であり、残る1つの入力が常に1である半加算器は削除し、半加算器の出力であるキャリアアウト端子の配線に0を供給させ、半加算器の出力であるサム端子の配線に1を供給させる、

2つの入力とも常に1である半加算器は削除し、半加算器の出力であるキャリアアウト端子の配線に1を供給させ、半加算器の出力であるサム端子の配線に0を供給させ、という変換の少なくとも一つを含んでいることを特徴とする請求項1に記載の論理回路圧縮方法。

【請求項6】上記特定の変換規則は、1つの入力が常に0であり、残る1つの入力が乗数、被乗数の値によって変化する半加算器は削除し、半加算器の出力であるキャリアアウト端子の配線に0を供給させ、半加算器の出力であるサム端子の配線に上記乗数、被乗数の値によって変化する値を供給させる、

1つの入力が常に1であり、残る1つの入力が乗数、被乗数の値によって変化する半加算器は削除し、半加算器の出力であるキャリアアウト端子の配線に上記乗数、被乗数の値によって変化する値を供給させ、半加算器の出力であるサム端子の配線に上記乗数、被乗数の値によって変化する値の反転値を供給せる、という変換の少なくとも一つを含んでいることを特徴とする請求項1に記載の論理回路圧縮方法。

【請求項7】上記特定の変換規則は、

単位遅延素子に定数が入力されているときは、その単位遅延素子を削除し、その単位遅延素子の出力端子に上記定数を供給せる、という変換を含んでいることを特徴とする請求項1に記載の論理回路圧縮方法。

【発明の詳細な説明】

【0001】

【産業上の利用分野】本発明は、ディジタル信号処理回路をLSI化する場合等に使用される論理回路圧縮方法に関するものである。

【0002】

【従来の技術】デジタル信号処理回路をLSI化する場合、論理設計とレイアウト設計の2段階により設計が行われる。

【0003】論理設計とは、アンドゲートやオアゲートやフルアダー（全加算器）やハーフアダー（半加算器）などと、それらゲートの接続をする配線の情報を持ったデータファイルを作成することである。一般的に、このデータファイルは、ネットリストと呼ばれている。ネットリストは、各半導体工場の専用のフォーマットのもの

$$\begin{array}{ll} \text{乗数} : X = (X_7, X_6, X_5, X_4, X_3, X_2, X_1, X_0) & 8\text{bit} \\ \text{被乗数} : Y = (Y_4, Y_3, Y_2, Y_1, Y_0) & 5\text{bit} \end{array}$$

の乗算をして、

$$\text{出力} : P = (P_{12}, P_{11}, P_{10}, P_9, P_8, P_7, P_6, P_5, P_4, P_3, P_2, P_1, P_0) \quad 13\text{bit}$$

を得るための計算方法である。

【0007】但し、

$A_i = Y_i$ と X_i のアンド（論理積）

$B_i = Y_i$ と X_i のアンド（論理積）

$C_i = Y_2$ と X_i のアンド（論理積）

$D_i = Y_1$ と X_i のアンド（論理積）

$E_i = Y_0$ と X_i のアンド（論理積） ($i = 0 \sim 7$)

である。

【0008】即ち、部分積 A_i 、 B_i 、 C_i 、 D_i 、 E_i を図9のAに示すように加算することにより、乗算結果 P_i を得ることが出来る。

【0009】さて図9のAの計算は全加算器や半加算器により計算していくことが可能であり、結局、論理設計は全加算器や半加算器、そしてそれらゲートの接続をする配線情報を作成することになる。

【0010】しかし上述の計算では、部分積 A_i 、 B_i 、 C_i 、 D_i 、 E_i は桁が1ビットずつシフトしており、規則的ではないため、ネットリスト作成は簡単でない（欠点1）。また仕様が変更されて、8ビット×5ビットではなく、8ビット×6ビットとなった時、最初からネットリストを書き直さなくてはいけないなどの欠点があった（欠点2）。

【0011】さらに、本願出願人が先に提案した回路記述合成装置（特願平3-195714号）では、パラメトリックに回路を記述できる装置について述べられているが、サイズ変換部での変換操作が難しい。なぜなら各部分積が1ビットずつシフトしており、規則的でなく、サイズ変換のアルゴリズムが複雑な為である（欠点3）。

【0012】これら欠点1、2、3は、部分積同士で規則性がないため起こる。そこでこれらの欠点を克服する方法として、図9のBの計算を行う事が考えられる。

【0013】図9のBの計算は、図9のAの部分積の空白個所に0を挿入して、13ビットのデータ

$$(0, 0, 0, 0, 0, A_7, A_6, A_5, A_4, A_3, A_2, A_1, A_0)$$

や、共通化されたフォーマット（例えば、EDIF、Verilog-HDL、VHDLなど）のものがある。

【0004】レイアウト設計とは、上述のネットリストのデータを受け取り、実際のLSIチップ上のどの位置にどのゲートやどの配線を置くかということを決める作業である。

【0005】論理設計について、さらに具体例を用いて説明する。

【0006】図9のAは、

$$\begin{array}{ll} \text{乗数} : X = (X_7, X_6, X_5, X_4, X_3, X_2, X_1, X_0) & 8\text{bit} \\ \text{被乗数} : Y = (Y_4, Y_3, Y_2, Y_1, Y_0) & 5\text{bit} \end{array}$$

$$\begin{array}{l} (0, 0, 0, 0, 0, B_7, B_6, B_5, B_4, B_3, B_2, B_1, B_0, 0) \\ (0, 0, 0, C_7, C_6, C_5, C_4, C_3, C_2, C_1, C_0, 0, 0) \\ (0, 0, D_7, D_6, D_5, D_4, D_3, D_2, D_1, D_0, 0, 0, 0) \\ (0, E_7, E_6, E_5, E_4, E_3, E_2, E_1, E_0, 0, 0, 0, 0) \end{array}$$

を部分積として、加算をする方法を取っている。これにより、加算すべき部分積が規則正しくなる。

【0014】この図9のBの計算をする回路図が図10である。図10の第1段の全加算器（FA）群により、3つの部分積、

$$\begin{array}{l} (0, 0, 0, 0, 0, A_7, A_6, A_5, A_4, A_3, A_2, A_1, A_0) \\ (0, 0, 0, B_7, B_6, B_5, B_4, B_3, B_2, B_1, B_0, 0) \\ (0, 0, 0, C_7, C_6, C_5, C_4, C_3, C_2, C_1, C_0, 0, 0) \end{array}$$

が加算され、第2段の全加算器（FA）群により、部分積、

$$(0, 0, D_7, D_6, D_5, D_4, D_3, D_2, D_1, D_0, 0, 0, 0)$$

が加算され、第3段の全加算器（FA）群により、さらに部分積、

$$(0, E_7, E_6, E_5, E_4, E_3, E_2, E_1, E_0, 0, 0, 0, 0)$$

が加算される。

【0015】第1～3段の全加算器（FA）群では、キャリセーブ方式で加算しているので、最後に、桁上げ加算をする必要があり、その計算を行うのが最終段の全加算器（FA）群である。

【0016】図10に対応したネットリストは、13×3個の全加算器（FA）群と、全加算器（FA）同士の接続をする配線情報、及び、13個の全加算器（FA）群より成る桁上げ加算器より構成される。

【0017】13×3個の全加算器(FA)群(第1～3段の全加算器(FA)群)が、規則正しく配列されていることが図10より明らかであり、当然、そのネットリストも規則正しい。従って、ネットリスト作成は簡単である(欠点1の克服)。

【0018】また仕様が変更されて、8ビット×5ビットではなく、8ビット×6ビットとなった時、ネットリストに、第1～3段の全加算器(FA)群と同一の13個の全加算器(FA)群を第4の全加算器(FA)群として追加するだけで良い(欠点2の克服)。

【0019】さらに回路記述合成装置のサイズ変換部での変換操作が簡単である。なぜなら、被乗数分の全加算器(FA)群だけを作成するという単純な操作で良いからである(欠点3の克服)。

【0020】このように、ダミーデータ0を挿入することにより、部分積に規則性を持たせて論理設計を容易化できる。

【0021】しかしこれから作られるネットリストを用いて、レイアウト設計を行うと、図10に示す回路と同じ構成の回路が、LSIチップ上に作られる。つまり0同士の加算を行うという無意味な全加算器などもチップ上に作られてしまい、回路規模が大きくなってしまうという欠点があった。

【0022】

【発明が解決しようとする課題】問題点は、従来の方法では論理設計を容易化するためなどにネットリスト中には冗長な回路が含まれていたことにより、実際に作られるLSIの回路規模が大きくなってしまうというものである。

【0023】

【課題を解決するための手段】本発明による第1の手段は、少なくとも1つの入力信号は定数である加算器若しくは単位遅延素子を含む冗長な回路構成のネットリストを入力(ステップ[1])とし、特定の変換規則(ステップ[2]及び[3])によって冗長でない回路構成のネットリストを出力(ステップ[4])するようにしたことを特徴とする論理回路圧縮方法である。

【0024】本発明による第2の手段は、上記特定の変換規則は、3つの入力とも常に0である全加算器(Full Adder)は削除し、全加算器の出力であるキャリアアウト(Co)端子の配線に0を供給させ、全加算器の出力であるサム(Sum)端子の配線に0を供給させる(図2のA)、2つの入力が常に0であり、残る1つの入力が常に1である全加算器(Full Adder)は削除し、全加算器の出力であるキャリアアウト(Co)端子の配線に0を供給させ、全加算器の出力であるサム(Sum)端子の配線に1を供給させる(図2のB)、1つの入力が常に0であり、残る2つの入力が常に1である全加算器(Full Adder)は削除し、全加算器の出力であるキャリアアウト(Co)端子の配線に1を供給させ、全加算器の出力であるサム(Sum)端子の配線に0を供給させ、全加算器の出力であるサム(Sum)端子の配線に1を供給させる(図2のC)、という変換の少なくとも一つを含んでいることを特徴とする第1の手段に記載の論理回路圧縮方法である。

(Sum)端子の配線に0を供給させる(図2のC)、3つの入力とも常に1である全加算器(Full Adder)は削除し、全加算器の出力であるキャリアアウト(Co)端子の配線に1を供給させ、全加算器の出力であるサム(Sum)端子の配線に1を供給させる(図2のD)、という変換の少なくとも一つを含んでいることを特徴とする第1の手段に記載の論理回路圧縮方法である。

【0025】本発明による第3の手段は、上記特定の変換規則は、2つの入力が常に0であり、残る1つの入力(in1)が乗数、被乗数の値によって変化する全加算器(Full Adder)は削除し、全加算器の出力であるキャリアアウト(Co)端子の配線に0を供給させ、全加算器の出力であるサム(Sum)端子の配線に上記乗数、被乗数の値によって変化する値を供給させる(図3のA)、1つの入力が常に0であり、もう1つの入力が常に1であり、残る1つの入力(in1)が乗数、被乗数の値によって変化する全加算器(Full Adder)は削除し、全加算器の出力であるキャリアアウト(Co)端子の配線に上記乗数、被乗数の値によって変化する値を供給させ、全加算器の出力であるサム(Sum)端子の配線に上記乗数、被乗数の値によって変化する値の反転値を供給させる(図3のB)、2つの入力が常に1であり、残る1つの入力(in1)が乗数、被乗数の値によって変化する全加算器(Full Adder)は削除し、全加算器の出力であるキャリアアウト(Co)端子の配線に1を供給させ、全加算器の出力であるサム(Sum)端子の配線に上記乗数、被乗数の値によって変化する値を供給させる(図3のC)、という変換の少なくとも一つを含んでいることを特徴とする第1の手段に記載の論理回路圧縮方法である。

【0026】本発明による第4の手段は、上記特定の変換規則は、1つの入力が常に0であり、残る2つの入力(in1、in2)が乗数、被乗数の値によって変化する全加算器(Full Adder)は削除し、半加算器(Half Adder)の2つの入力端子に上記乗数、被乗数の値によって変化する2つの値を供給させ、全加算器の出力であるキャリアアウト(Co)端子の配線に上記半加算器の出力であるキャリアアウトを供給させ、全加算器の出力であるサム(Sum)端子の配線に上記半加算器の出力であるサムを供給させる(図4のA)、1つの入力が常に1であり、残る2つの入力(in1、in2)が乗数、被乗数の値によって変化する全加算器(Full Adder)は削除し、全加算器の出力であるキャリアアウト(Co)端子の配線に上記乗数、被乗数の値によって変化する2つの値の論理和(オア回路)を供給させ、全加算器の出力であるサム(Sum)端子の配線に上記乗数、被乗数の値によって変化する2つの値の排他的否定論理和(イクスクルーシブノア回路)を供給させる(図4のB)、という変換の少なくとも一つを含んでいることを特徴とする第1の手段に記載の論理回路圧縮方法である。

【0027】本発明による第5の手段は、上記特定の変

換規則は、2つの入力とも常に0である半加算器(Half Adder)は削除し、半加算器の出力であるキャリアアウト(Co)端子の配線に0を供給させ、半加算器の出力であるサム(Sum)端子の配線に0を供給させる(図5のA)、1つの入力が常に0であり、残る1つの入力が常に1である半加算器(Half Adder)は削除し、半加算器の出力であるキャリアアウト(Co)端子の配線に0を供給させ、半加算器の出力であるサム(Sum)端子の配線に1を供給させる(図5のB)、2つの入力とも常に1である半加算器(Half Adder)は削除し、半加算器の出力であるキャリアアウト(Co)端子の配線に1を供給させ、半加算器の出力であるサム(Sum)端子の配線に0を供給させる(図5のC)、という変換の少なくとも一つを含んでいることを特徴とする第1の手段に記載の論理回路圧縮方法である。

【0028】本発明による第6の手段は、上記特定の変換規則は、1つの入力が常に0であり、残る1つの入力(in1)が乗数、被乗数の値によって変化する半加算器(Half Adder)は削除し、半加算器の出力であるキャリアアウト(Co)端子の配線に0を供給させ、半加算器の出力であるサム(Sum)端子の配線に上記乗数、被乗数の値によって変化する値を供給させる(図6のA)、1つの入力が常に1であり、残る1つの入力(in1)が乗数、被乗数の値によって変化する半加算器(Half Adder)は削除し、半加算器の出力であるキャリアアウト(Co)端子の配線に上記乗数、被乗数の値によって変化する値を供給させ、半加算器の出力であるサム(Sum)端子の配線に上記乗数、被乗数の値によって変化する値の反転値を供給させる(図6のB)、という変換の少なくとも一つを含んでいることを特徴とする第1の手段に記載の論理回路圧縮方法である。

【0029】本発明による第7の手段は、上記特定の変換規則は、単位遅延素子に定数が入力されているときは、その単位遅延素子を削除し、その単位遅延素子の出力端子に上記定数を供給させる(図7)、という変換を含んでいることを特徴とする第1の手段に記載の論理回路圧縮方法である。

【0030】

【作用】これによれば、冗長な回路構成のネットリストから冗長な回路を削除したネットリストを作ることができ、このような新たなネットリストからレイアウト設計することにより回路規模の小さいLSIを実現することができる。

【0031】

【実施例】本発明の論理回路圧縮方法のフローチャートの例を図1に示す。

【0032】すなわちこの図において、ステップ【1】で冗長な回路のネットリストが読み込まれる。次にステップ【2】でそのネットリスト内に後述する特定の冗長な回路があるかないかが判断される。ここでもし冗長な

回路があればステップ【3】で後述する変換規則に従つて冗長な回路が削除される。

【0033】さらにこのステップ【2】及び【3】が冗長な回路がなくなるまで繰り返し実行される。そしてステップ【2】でそのネットリスト内にある冗長な回路がなくなると、ステップ【4】で冗長でない回路のネットリストが outputされる。

【0034】こうして上述の方法によれば、冗長な回路構成のネットリストから冗長な回路を削除したネットリストを作ることができ、このような新たなネットリストからレイアウト設計することにより回路規模の小さいLSIを実現することができるものである。

【0035】さらに上述の方法で、冗長な回路の有無を判断して削除するための特定の変換規則には、以下のよ

うなものが用いられる。

【0036】すなわち図2のAは、3つの入力とも常に0である全加算器(Full Adder)の場合であって、この場合にはこの全加算器は削除し、同図の右側に示すように全加算器の出力であるキャリアアウト(Co)端子の配線に0を供給させ、全加算器の出力であるサム(Sum)端子の配線に0を供給させる。

【0037】また図2のBは、2つの入力が常に0であり、残る1つの入力が常に1である全加算器(Full Adder)の場合であって、この場合にはこの全加算器は削除し、同図の右側に示すように全加算器の出力であるキャリアアウト(Co)端子の配線に0を供給させ、全加算器の出力であるサム(Sum)端子の配線に1を供給させる。

【0038】また図2のCは、1つの入力が常に0であり、残る2つの入力が常に1である全加算器(Full Adder)の場合であって、この場合にはこの全加算器は削除し、同図の右側に示すように全加算器の出力であるキャリアアウト(Co)端子の配線に1を供給させ、全加算器の出力であるサム(Sum)端子の配線に0を供給させる。

【0039】また図2のDは、3つの入力とも常に1である全加算器(Full Adder)の場合であって、この場合にはこの全加算器は削除し、同図の右側に示すように全加算器の出力であるキャリアアウト(Co)端子の配線に1を供給させ、全加算器の出力であるサム(Sum)端子の配線に1を供給させる。

【0040】さらに図3のAは、2つの入力が常に0であり、残る1つの入力(in1)が乗数、被乗数の値によって変化する全加算器(Full Adder)の場合であって、この場合にはこの全加算器は削除し、同図の右側に示すように全加算器の出力であるキャリアアウト(Co)端子の配線に0を供給させ、全加算器の出力であるサム(Sum)端子の配線に上記乗数、被乗数の値によって変化する値を供給させる。

【0041】また図3のBは、1つの入力が常に0であり、もう1つの入力が常に1であり、残る1つの入力(in1)が乗数、被乗数の値によって変化する全加算器

(FullAdder)の場合であって、この場合にはこの全加算器は削除し、同図の右側に示すように全加算器の出力であるキャリアウト (Co) 端子の配線に上記乗数、被乗数の値によって変化する値を供給させ、全加算器の出力であるサム (Sum) 端子の配線に上記乗数、被乗数の値によって変化する値の反転値を供給させる。

【0042】また図3のCは、2つの入力が常に1であり、残る1つの入力 (in1) が乗数、被乗数の値によって変化する全加算器(Full Adder)の場合であって、この場合にはこの全加算器は削除し、同図の右側に示すように全加算器の出力であるキャリアウト (Co) 端子の配線に1を供給させ、全加算器の出力であるサム (Sum) 端子の配線に上記乗数、被乗数の値によって変化する値を供給させる。

【0043】さらに図4のAは、1つの入力が常に0であり、残る2つの入力 (in1, in2) が乗数、被乗数の値によって変化する全加算器(Full Adder)の場合であって、この場合にはこの全加算器は削除し、同図の右側に示すように半加算器(Half Adder)の2つの入力端子に上記乗数、被乗数の値によって変化する2つの値を供給させ、全加算器の出力であるキャリアウト (Co) 端子の配線に上記半加算器の出力であるキャリアウトを供給させ、全加算器の出力であるサム (Sum) 端子の配線に上記半加算器の出力であるサムを供給させる。

【0044】また図4のBは、1つの入力が常に1であり、残る2つの入力 (in1, in2) が乗数、被乗数の値によって変化する全加算器(Full Adder)の場合であって、この場合にはこの全加算器は削除し、同図の右側に示すように全加算器の出力であるキャリアウト (Co) 端子の配線に上記乗数、被乗数の値によって変化する2つの値の論理和 (オア回路) を供給させ、全加算器の出力であるサム (Sum) 端子の配線に上記乗数、被乗数の値によって変化する2つの値の排他的否定論理和 (イクス・クルーシブノア回路) を供給させる。

【0045】さらに図5のAは、2つの入力とも常に0である半加算器(Half Adder)の場合であって、この場合にはこの半加算器は削除し、同図の右側に示すように半加算器の出力であるキャリアウト (Co) 端子の配線に0を供給させ、半加算器の出力であるサム (Sum) 端子の配線に0を供給させる。

【0046】また図5のBは、1つの入力が常に0であり、残る1つの入力が常に1である半加算器(Half Adder)の場合であって、この場合にはこの半加算器は削除し、同図の右側に示すように半加算器の出力であるキャリアウト (Co) 端子の配線に0を供給させ、半加算器の出力であるサム (Sum) 端子の配線に1を供給させる。

【0047】また図5のCは、2つの入力とも常に1である半加算器(Half Adder)の場合であって、この場合にはこの半加算器は削除し、同図の右側に示すように半加算器の出力であるキャリアウト (Co) 端子の配線に1を

供給させ、半加算器の出力であるサム (Sum) 端子の配線に0を供給させる。

【0048】さらに図6のAは、1つの入力が常に0であり、残る1つの入力 (in1) が乗数、被乗数の値によって変化する半加算器(Half Adder)の場合であって、この場合にはこの半加算器は削除し、同図の右側に示すように半加算器の出力であるキャリアウト (Co) 端子の配線に0を供給させ、半加算器の出力であるサム (Sum) 端子の配線に上記乗数、被乗数の値によって変化する値を供給させる。

【0049】また図6のBは、1つの入力が常に1であり、残る1つの入力 (in1) が乗数、被乗数の値によって変化する半加算器(Half Adder)の場合であって、この場合にはこの半加算器は削除し、同図の右側に示すように半加算器の出力であるキャリアウト (Co) 端子の配線に上記乗数、被乗数の値によって変化する値を供給させ、半加算器の出力であるサム (Sum) 端子の配線に上記乗数、被乗数の値によって変化する値の反転値を供給させる。

【0050】さらに図7は、単位遅延素子 (フリップフロップFF) に定数が入力されている場合であって、この場合には、その単位遅延素子を削除し、同図の右側に示すようにその単位遅延素子の出力端子に上記定数を供給させる。

【0051】これによって、例えば上述の図10に示した冗長な回路構成のネットリストから例えば図8に示すような冗長な回路を削除したネットリストを作ることができ、このような新たなネットリストからレイアウト設計することにより回路規模の小さいLSIを実現することができる。

【0052】なお上述の例では、いわゆるキャリセーブ型の乗算回路について説明をしたが、本発明はこの他にも例えばWallace Tree (ワラス・ツリー) 型で部分積を加算していく乗算回路や、さらに乗算回路以外の回路についても適用できる。

【0053】また本発明の論理回路圧縮方法によれば、以下に挙げる利点がある。即ち、論理設計の段階では設計容易化の為に規則性を重んじて冗長な回路が含まれた形で設計していきネットリストを作成する。そのネットリストに対して本発明の論理圧縮プログラムを適用して冗長でない回路のネットリストに変換する。そしてこの変換されたネットリストによりレイアウト設計を行う。これにより論理設計は容易になるとともに、冗長でない回路のレイアウトを行えるので実際のLSIの回路規模も小さく済むものである。

【0054】

【発明の効果】この発明によれば、冗長な回路構成のネットリストから冗長な回路を削除したネットリストを作ることができ、このような新たなネットリストからレイアウト設計することにより回路規模の小さいLSIを実

現することができるようになった。

【図面の簡単な説明】

【図1】本発明による論理回路圧縮方法の一例のフローチャート図である。

【図2】本発明による特定の変換規則の説明のための図である。

【図3】本発明による特定の変換規則の説明のための図である。

【図4】本発明による特定の変換規則の説明のための図である。

【図5】本発明による特定の変換規則の説明のための図である。

【図6】本発明による特定の変換規則の説明のための図である。

【図7】本発明による特定の変換規則の説明のための図

である。

【図8】本発明を用いて冗長な回路を削除した部分積加算回路の一例の構成図である。

【図9】乗算計算の説明のための図である。

【図10】従来の部分積加算回路の構成図である。

【符号の説明】

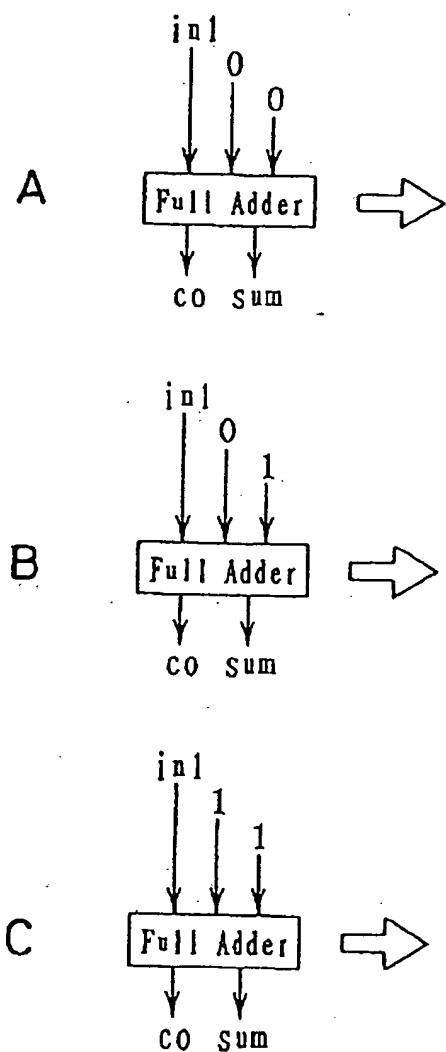
〔1〕冗長な回路のネットリストの読み込みのステップ

〔2〕ネットリスト内の特定の冗長な回路の有無の判断のステップ

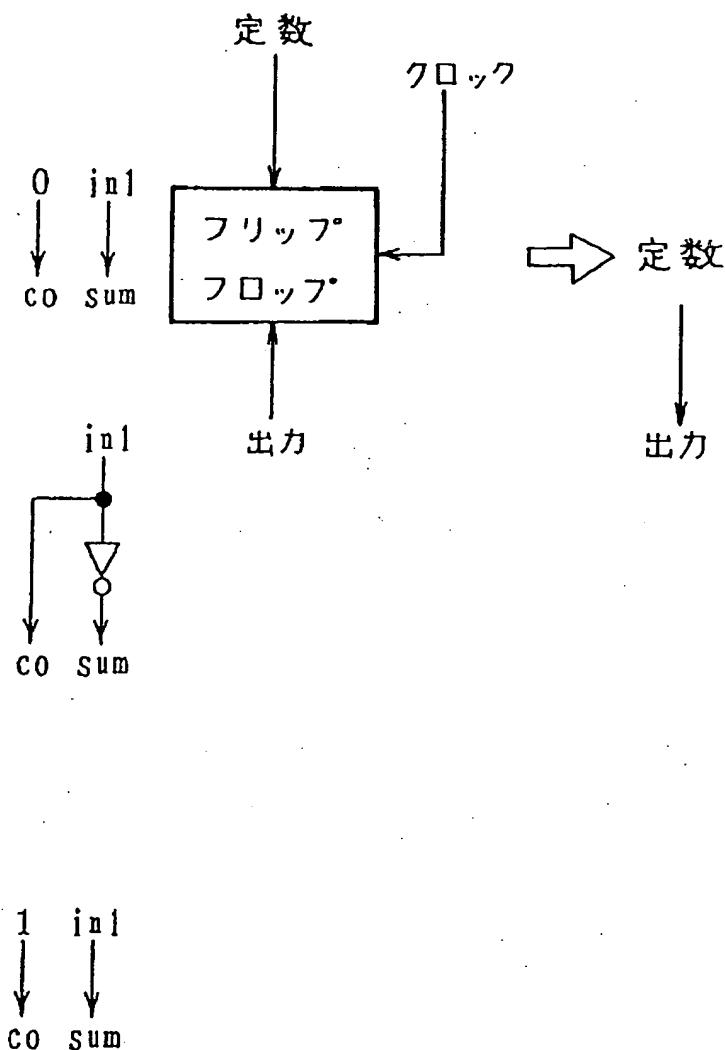
〔3〕変換規則に従って冗長な回路を削除するステップ

〔4〕冗長でない回路のネットリストを出力するステップ

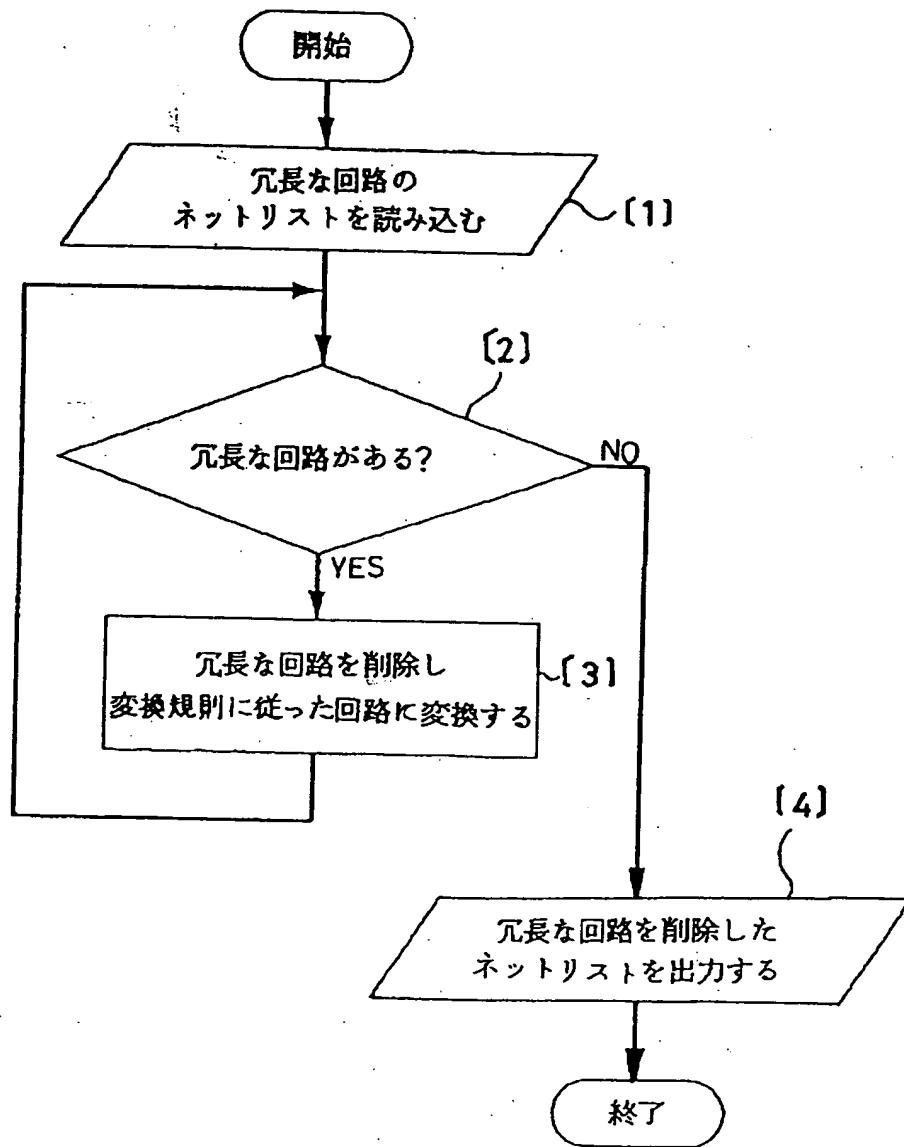
【図3】



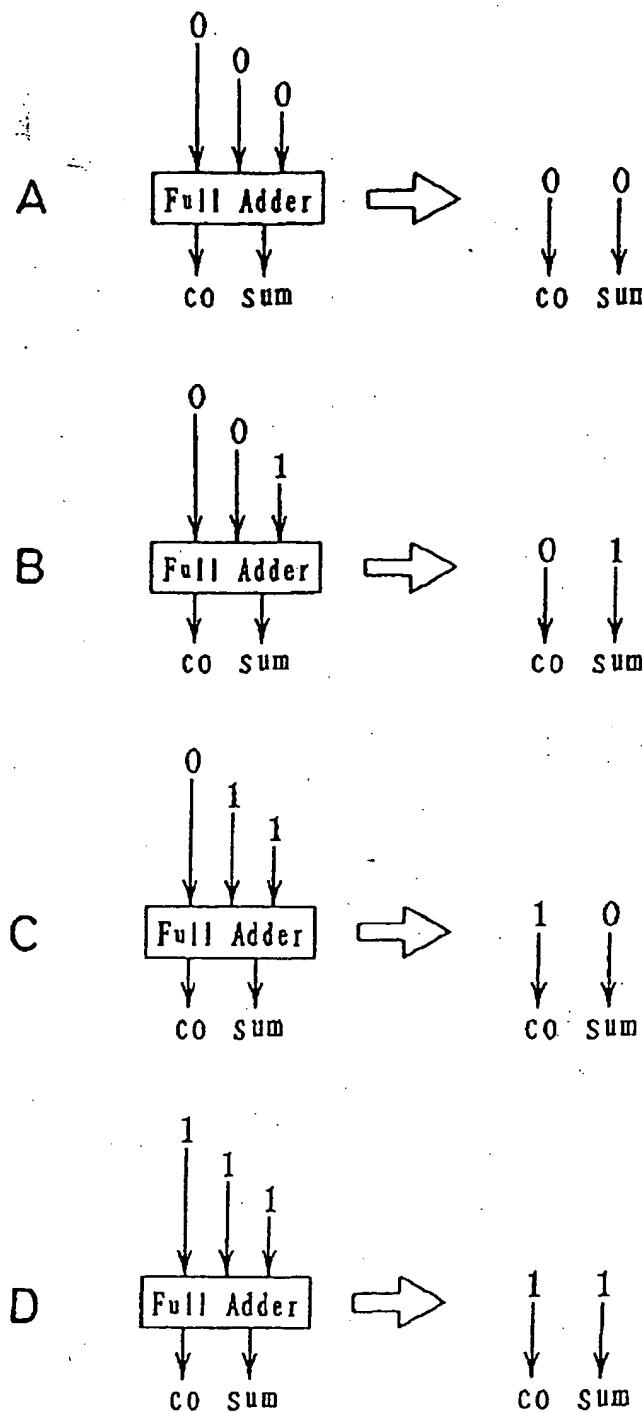
【図7】



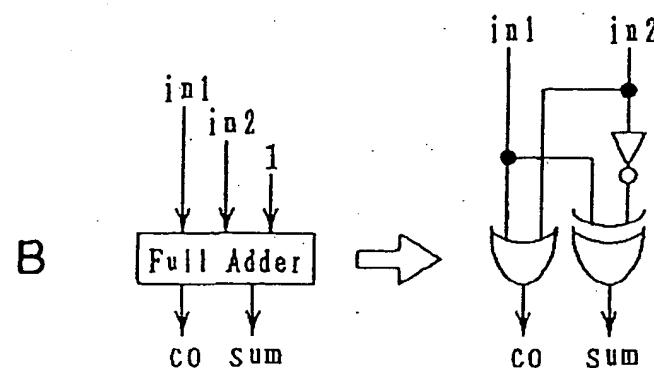
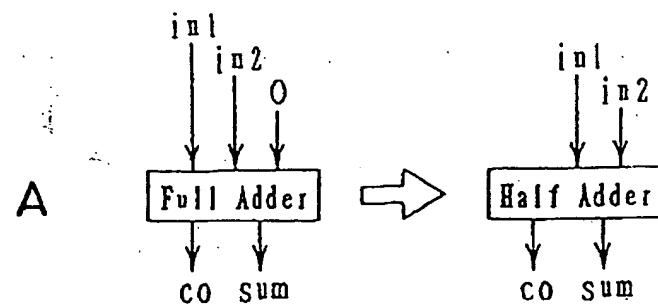
【図1】



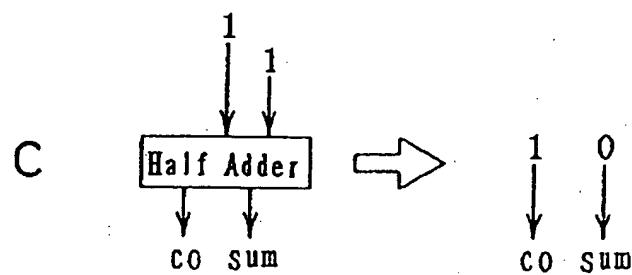
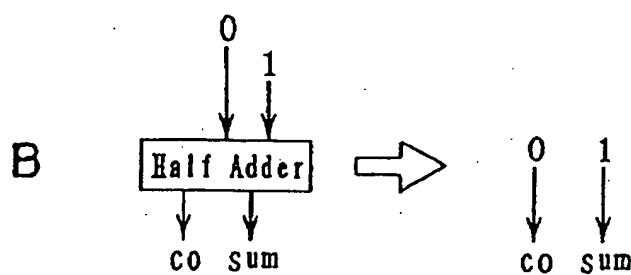
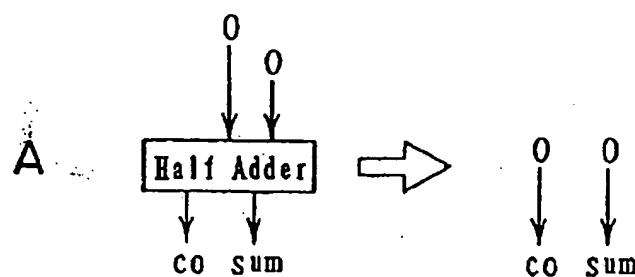
【図2】



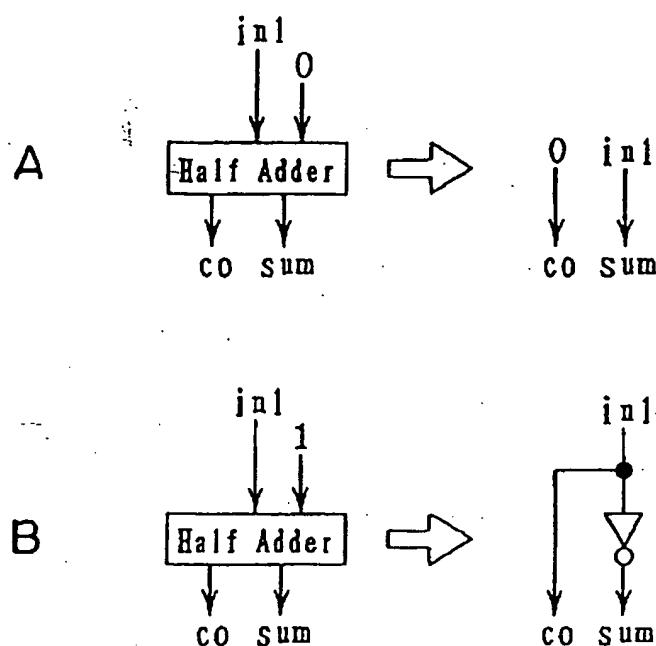
【図4】



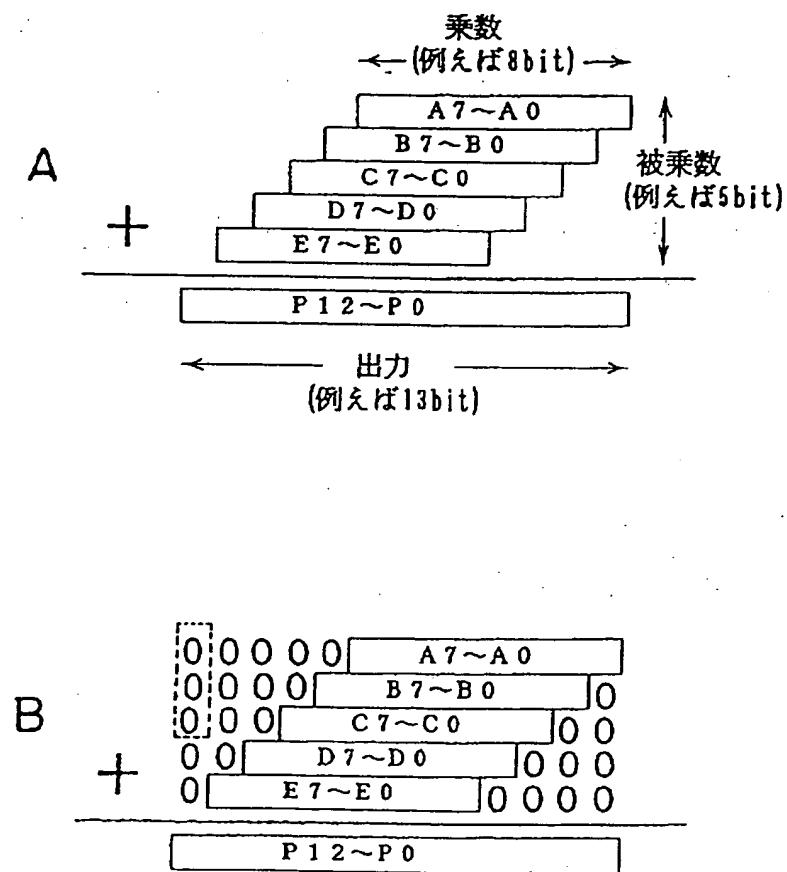
【図5】



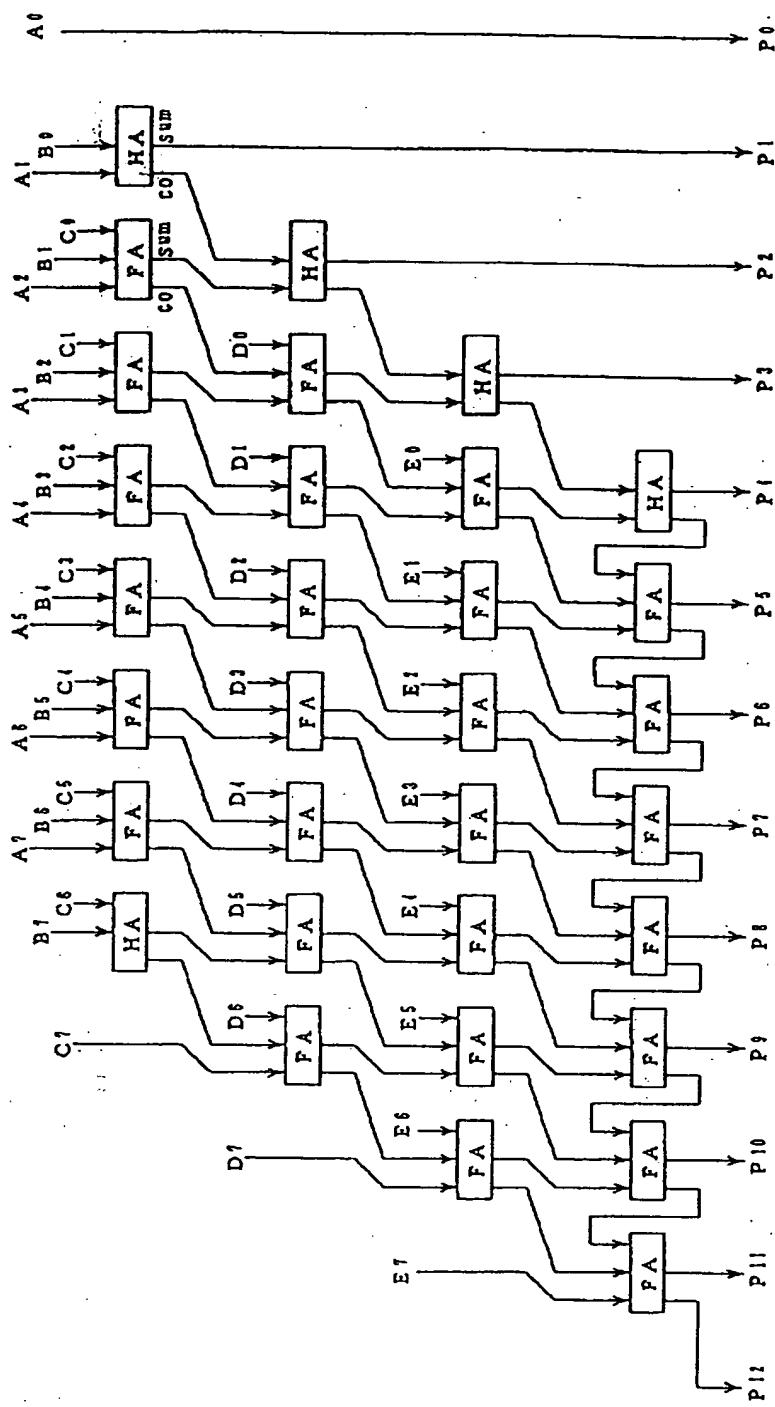
【図6】



【図9】



【图8】



【図10】

